Digital signal processor control of scanned probe microscopes

David R. Baselt[a)]
Division of Chemistry, California Institute of Technology, Pasadena, California 91125

Steven M. Clark
Division of Biology, California Institute of Technology, Pasadena, California 91125

Michael G. Youngquist
Division of Chemistry, California Institute of Technology, Pasadena, California 91125

Charles F. Spence
Division of Biology, California Institute of Technology, Pasadena, California 91125

John D. Baldeschwieler
Division of Chemistry, California Institute of Technology, Pasadena, California 91125

(Received 20 March 1992; accepted for publication 17 March 1993)

Digital signal processors have made it possible to control scanned probe microscopes using straightforward software emulations of analog circuits. Using a system consisting of a commercially available digital signal processor board interfaced to analog I/O, we have developed algorithms for self-optimizing feedback, raster generation (with hysteresis correction, sample tilt compensation, and scan rotation), lock-in detection, and automatic tip-sample approach. We also discuss an instruction parser that takes advantage of the digital architecture to allow automatic operation for extended periods.

I. INTRODUCTION

Most commercial manufacturers of scanned-probe microscopes (SPMs) use digital signal processor (DSP)-based control systems to drive their instruments. Such control systems provide a combination of versatility and performance not readily available from other system architectures. However, noncommercial SPMs have generally used other architectures at least in part because of the complex programming required for the DSP-based design.

Recently, though, increased DSP computing power has enabled the use of more straightforward digital control algorithms. At the same time DSPs and other associated hardware have become more widespread and less expensive. Using state-of-the-art technology, we have developed a number of simple control algorithms that make it possible to construct high-performance DSP-based systems with minimal effort. Such systems can rival or even exceed commercial instruments in performance while costing far less. More importantly, they allow free experimentation with SPM design.

A number of articles in the literature[1–11] describe analog and digital control systems. While these concentrate on hardware issues, this paper will focus on software: in particular, emulations of analog circuits with simple yet high-performance digital algorithms for feedback, raster generation, and modulation imaging. We will provide only background information on our system hardware, which we have already described in detail.[12]

We are currently using the DSP-based control system described here for scanning tunneling microscopy (STM), atomic force microscopy (AFM), near-field scanning optical microscopy, and low-temperature tunneling spectroscopy. The system also suits most other scanned-probe experiments.[13]

II. THE DSP-BASED ARCHITECTURE

An SPM consists of a sensor that measures some local property of a sample, and a scanner that can move the sensor in three dimensions. The control system positions the sensor (by applying voltages) and records its measurements (by reading voltage outputs), thereby imaging the measured property.

In the DSP-based control system (Fig. 1), a microprocessor—the DSP—controls via digital-to-analog converters (DACs) the voltages applied to the SPM stage, and monitors via analog-to-digital converters (ADCs) the sensor output voltages. The software on the DSP carries out all control functions.

Although a number of different microprocessors could control SPMs, DSPs have two significant advantages: (1) they are designed for real-time processing of audiofrequency signals similar to those used in SPM; and (2) they can be bought in convenient board-level products and programmed using high-level languages such as C. The DSP board plugs into a host personal computer (PC) and serves as a self-contained computer dedicated to SPM control.

A. Hardware components

DSP-based control systems have two main components, the DSP board and its analog I/O.

Analog I/O. For optimal performance, we have constructed special low-noise analog I/O.[12] Given the substantial investment in time this requires, others may prefer to use commercial I/O instead.[14] However, note that many commercial I/O products are intended for prototyping purposes only and have noise levels unsuitable for SPM.

[a)]Current address: Department of Biology, Yale University, P.O. Box 6666, New Haven, CT 06511.

 0034-6748/93/64(7)/1874/9/$6.00

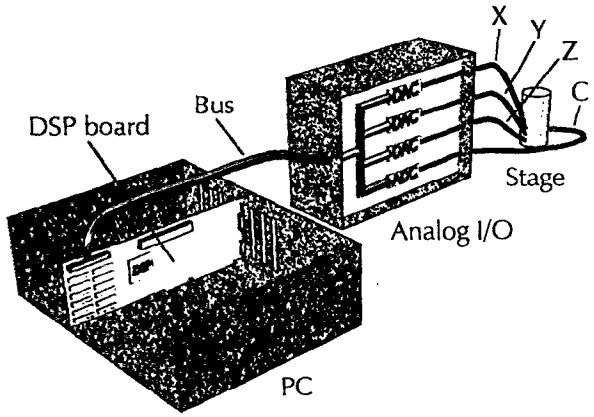

FIG. 1. DSP-based SPM hardware. The DSP board plugs into a PC and communicates with an analog I/O box that resides outside the PC. For reasons of simplicity the diagram omits the signal and high voltage amplifiers between the analog I/O and the stage. The diagram also omits several ADCs and DACs; our electronics actually have four ADCs and eight DACs.

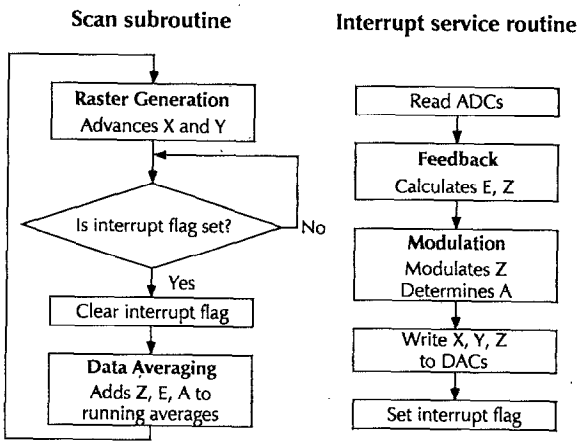

FIG. 2. Software algorithms used during one timer period while scanning. Every 10 μs the DSP timer triggers an interrupt, causing an automatic jump to the interrupt service routine. This jump should occur during the waiting loop shown. After the DSP completes the interrupt service routine, it resumes the scan subroutine. Not shown: data storage (occurs less than once per timer period), tip-sample approach (not used while scanning), and user interface (independent of timer period). *E*: error signal. *X*,*Y*,*Z*: scanner voltages. *A*: modulation amplitude.

DSP board. Commerical vendors now offer a wide range of DSP boards suited to SPM control. We use the TMS320C30 processor board from Spectrum Signal Processing,[15] based on the Texas Instruments DSP of the same name.[16] This board has three features that greatly simplify programming: its floating point architecture simplifies development of control algorithms, while its dual-port memory simplifies PC-DSP communications, and its onboard memory suffices to store several images if acquisition rates exceed the maximum PC-DSP data transfer rate.

III. THE ALGORITHMS

The software handles all instrument control functions: feedback, *XY* raster generation, data acquisition, and tip-sample approach. In addition, it provides a user interface, taking instructions from the user and displaying data.

The DSP-based architecture lends itself to a division of tasks in which the DSP handles instrument control functions and the PC handles the user interface. Below we describe the DSP algorithms, then briefly discuss a user interface enhancement we have found useful, the programmable parser. First, though, we introduce the timing loop upon which all the algorithms depend.

A few notes on terminology: *Z* refers to the voltage that moves the probe toward or away from the sample, *X* to the fast raster voltage, and *Y* to the slow raster voltage. The "sensor output" *C* is an ADC reading proportional to the probe-sample separation (i.e., tunneling current in STM, cantilever deflection in AFM).

A. Timing considerations

Most SPM control algorithms require consistent and predictable timing to function properly. The characteristics of a digital feedback loop, for example, vary with its sampling rate. Likewise, an SPM needs a known and constant scan rate in order to predict and control effects of thermal drift, feedback characteristics, and scanner creep. We use a timer built-in to the DSP to ensure consistent timing of our control algorithms.

The timer period, usually 10 μs, serves as the fundamental clock of the SPM. Once per timer period, the DSP advances the *XY* raster, reads the ADCs, performs the feedback loop calculations, and stores image data (Fig. 2); then it waits for the timer to trigger again.

The timer period limits the number of features that the DSP can handle simultaneously. A 10 μs period corresponds to 166 instruction cycles on our DSP and suffices for most experiments; for complex experiments we have to increase the period to 15 μs. Whenever possible, though, we prefer the shorter timer period because it permits faster scans and more oversampling on slow scans. Also, the optimum response rate of the feedback (its ability to follow features on the sample surface) depends in part on the timer period. The ADC conversion time, 5–10 μs for 16 bit ADCs, places a lower bound on the timer period.

B. Self-optimizing proportional-integral-differential feedback

By continually adjusting *Z*, the feedback loop attempts to maintain the sensor output voltage at a constant level (the "setpoint") specified by the user. The computing power of DSPs permits software emulation of analog feedback, but digital feedback surpasses analog feedback in its abilities to support a wide range of feedback parameters, simplify implementation of features such as spectroscopy and digital modulation, and permit self-regulating feedback.

1. PID feedback

Our system uses a proportional-integral-differential (PID) feedback loop, a common algorithm useful for a wide variety of applications.[17] PID feedback sums the outputs of three separate feedback loops (Table I). Propor-

TABLE I. Proportional, trapezoidal-approximation integral, derivative, and PID feedback algorithms.

Proportional	${}^{P}Z_t = PE_t$
Integral	${}^{I}Z_t = IZ_{t-1} + I(E_t + E_{t-1})$
Derivative	${}^{D}Z_t = D(E_t - E_{t-1})$
PID	$Z_t = {}^{P}Z_t + {}^{I}Z_t + {}^{D}Z_t$

tional feedback responds quickly to small features but cannot maintain a dc offset. Integral feedback maintains a dc offset but cannot respond to small features without oscillating. Derivative feedback reduces overshoot and oscillations, but amplifies high frequency (about half the sampling frequency) noise.

An operational formula for PID feedback follows:

$$Z_t = Z_{t-1} + aE_t + bE_{t-1} + cE_{t-2},$$

where

$$E_t = C_t - S$$

and t=time in timer periods, $Z_t = Z$ at time t, E_t=error signal at time t, C=ADC reading at time t, S=setpoint, $a = P + I + D$, $b = -P + I - 2D$, $c = D$, P=proportional feedback gain, I=integral feedback gain, D=derivative feedback gain.

The user sets P, I, D, and S, from which the PC software determines a, b, and c. The feedback gains control the response rate of the feedback: when they are too low the feedback responds slowly and features wash out; when too high the feedback overreacts and oscillates. In our experience with AFM and STM, setting P 10–100 times higher than I provides optimum feedback. Derivative feedback has limited use in SPM due to its tendency to amplify high frequency noise. Setting D about ten times greater than I does reduce overshoot, but only to a small extent. Negative values of D have the reverse and sometimes useful effect of decreasing high-frequency noise but increasing overshoot. Those who find processor time limited may wish to save a few instruction cycles by removing derivative feedback.

2. Optimizing feedback gains

Adjusting P, I, and D in the usual manner, by inspecting image quality, generally fails to produce optimal instrument performance. Step response curves provide a better visual indicator of feedback rise time, noise level, and oscillations.

To generate a step response curve, the DSP changes the setpoint by an amount ΔS and records the error signal for a few milliseconds afterward while the feedback adjusts to the new setpoint. The feedback behaves the same during a step response measurement as it would if the tip encountered a step on the sample and both the tip and step were perfectly sharp.

Using step response curves, the user can optimize feedback gains either manually or automatically.

Manual optimization. For manual feedback optimization, the user adjusts P, I, and D based on the appearance of step response curves displayed after each adjustment (Fig. 3). On our AFMs and STMs, the optimized feedback

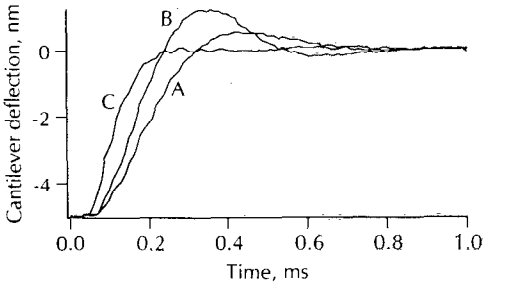

FIG. 3. Manual optimization of AFM feedback using step response curves. $\Delta S = 50$ Å. (A) Integral feedback only: $P=0$, $I=0.002$, $D=0$. (B) Raising I to 0.003 increases the response rate but also increases overshoot (or oscillations). (C) Raising P to 0.03 increases response and decreases overshoot. The feedback now settles in about 0.25 ms—within one data point if a scan has 250 points/line acquired at 8 lines/s or slower. These step response curves were acquired on our 2 μm range scanned-cantilever AFM.

usually settles within 20–50 feedback cycles, corresponding to one data point or less if the scan has 250 points/line acquired no faster than 10 lines/s. Thus, the feedback follows topography with a lag of no more than one data point. When we optimize the feedback in this manner we cannot detect any difference between a slope of topography (Z) image and an error signal (E) image (Fig. 4).

Automatic optimization. Control systems engineers have developed a number of adaptive feedback algorithms that eliminate the need for manual adjustment of feedback parameters.[18] These algorithms use mathematical models of the stage and are therefore instrument-specific. We have developed a general-purpose "evolutionary" algorithm that optimizes feedback parameters while making a minimal number of assumptions about the nature of the stage. We typically use this algorithm after installing a new tip or sample and when oscillations or other signs of poor feedback performance appear. The software generates nine "mutations" of the "parent" gain factors P, I, and D according to the following example:

$${}^{i}P_g = {}^{\text{opt}}P_{g-1}\,{}^{i}R,$$

where ${}^{i}P_g$=mutation i of proportional gain, $i=1$ to 9, ${}^{\text{opt}}P_{g-1}$=parent proportional gain (optimal mutation of previous generation), g=generation, ${}^{i}R$=a random number between $(1+V_g)$ and $1/(1+V_g)$, and V_g ("variance") =initially between 0.2 and 0.6. Note that the algorithm multiplies the three parameters ${}^{\text{opt}}P$, ${}^{\text{opt}}I$, and ${}^{\text{opt}}D$ by three *different* random numbers. The DSP generates ten step response curves (five in which it increases the setpoint, and five in which it decreases the setpoint) for the parent and each mutation, and calculates an optimization index for each curve.

$$O = oE_{\max} + \frac{\Sigma_{t=0}^{t_{\max}} |E_t|}{t_{\max}}, \tag{1}$$

where O=optimization index, $E_{\max}$=overshoot, o=user-specified overshoot factor (typically 0.3), and $t_{\max}$ =number of points in step response curve (typically 100).

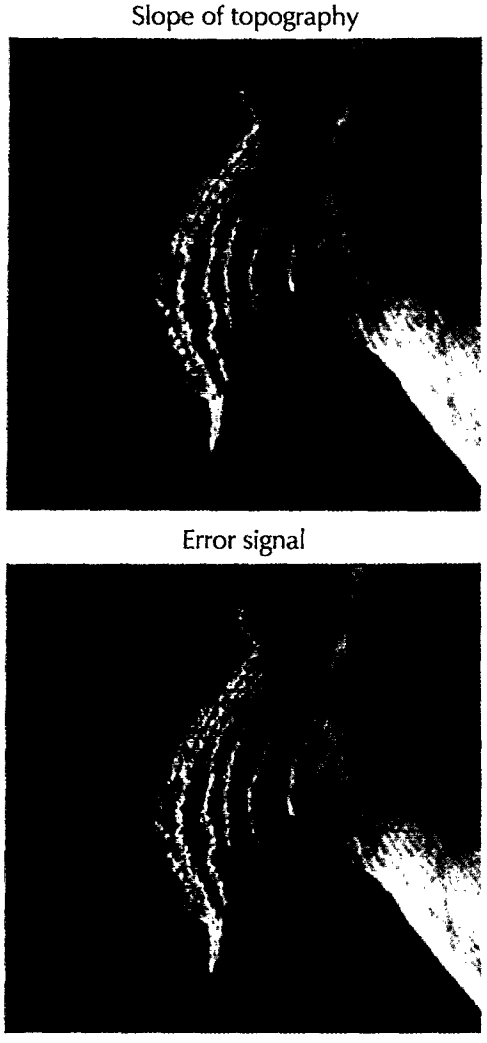

FIG. 4. Comparison of slope of topography (dZ/dX) and error signal images. Since we have adjusted the step response curve to settle within one data point, the two images differ negligibly. The sample is probably a fragment of an electron-beam-deposited AFM tip. The substrate is float-polished quartz (image size 385×385 nm).

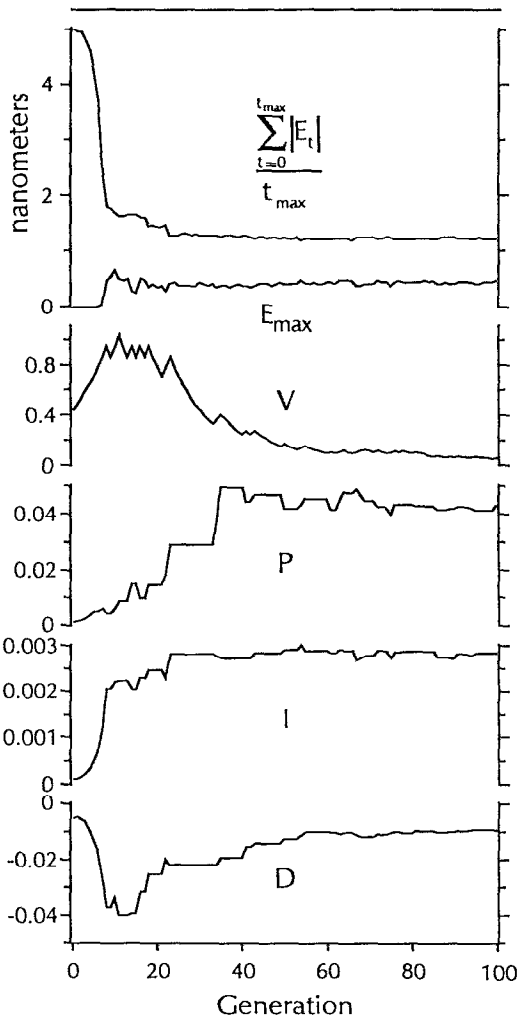

FIG. 5. Evolutionary optimization of feedback parameters using step response curves. The optimization ran on a 2 μm scan range AFM using a 50 Å step size and 100 points per step response curve. The software took six step response curves for each mutation. Total elapsed time, 10 s.

The feedback gains of the best mutation replace the parent. We define the best mutation as the one that yields the lowest average optimization index.

After each generation that a mutation replaces the parent, the software raises the variance V by 10%. Conversely, each generation that the software fails to find a mutation better than the parent, it lowers the variance 10%. V starts at 0.4; when it passes a preset threshold, usually 0.2, the optimization process terminates.

The optimizer usually cannot recover from oscillating feedback, so the user should initially set feedback parameters lower than optimal. A special mechanism guards against occasions when the optimizer itself causes the feedback to go into oscillation. If the best optimization index increases by more than 150% over one generation, the PC turns the feedback off for 10 ms and reduces P, I, and D by 50%.

Figure 5 demonstrates the optimization process. For this demonstration we have used our 2 μm range scanned-cantilever AFM.[19] The graphs show standard deviation, overshoot, variance, and feedback gains over time. We set the feedback gains unusually low at the beginning to better demonstrate the progress of optimization. We have also set D negative; when we set D positive the optimizer only lowers it until it becomes insignificant. Normally the algorithm would terminate when the variance falls below 0.2 (generation 40), but for the illustration we have disabled this feature. The time required for all 100 generations was 10 s.

Figure 6 shows step response curves taken before and after the optimization process of Fig. 5. Note that after optimizing, the error signal first overshoots then undershoots zero. Increasing the overshoot factor o reduces overshoot but tends to increase undershoot.

The automatic optimizer is most useful on long-range instruments that tend to oscillate, since it can maximize response rate while minimizing oscillations. On more stable instruments an experienced user generally does a better job of optimizing the feedback, but requires much more time. In this case, we sometimes manually adjust the feed-

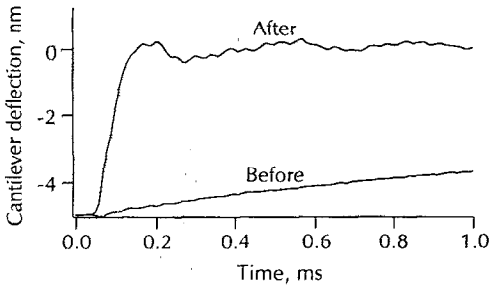

FIG. 6. Step response curves taken before and after the automatic optimization shown in Fig. 5.

back after automatic optimization, a process that still takes much less time than fully manual optimization.

Evolutionary optimization works better on AFMs than on STMs, since STMs generally produce poor step response curves. We suspect that the logarithmic dependence of tunneling current on tip-sample separation may be responsible, but have not tried using a log amp to linearize the current.

AFMs with higher noise levels may benefit from more step response curves per mutation, a higher termination variance, or fewer mutations per generation. Greater noise translates into more uncertainty in the optimization index O, so the optimizer more often finds mutations that appear to perform better than a fully optimized parent. The most direct solution is to average more step response curves together, thereby reducing the noise. However, the optimization process will then take longer. Alternatively, the programmer can lower the standard that defines "optimized feedback" either by increasing the termination variance or decreasing the number of mutations per generation (reducing the chance that the optimizer will find an apparently better mutation).

C. Scanning

The scan subroutine generates a rotatable XY raster and stores incoming data. The subroutine can "tilt" the scan and correct for hysteresis.

1. The basic algorithm

An XY raster ramps X repeatedly between selected minimum and maximum values, advancing Y after each cycle. It may also rotate the scan direction, a useful feature easy to implement on DSP-based systems. As described below, we emulate a continuous analog raster as closely as possible by averaging incoming data over several timer periods.

Continuous raster. To achieve the user-specified scan rate, the raster algorithm adjusts the number of timer periods spent at each data point. To keep scanner motion as continuous as possible, the raster advances once per period rather than once per data point (20–50 timer periods typically comprise one data point). Further, the data recorded for each point represent an average value over the timer periods making up that point. The combination of continuous scanner motion and data averaging ensures that features somewhat smaller than the distance between data points still appear in the image. In addition, data averaging improves the signal-to-noise ratio.

The timer period determines the fastest possible scan rate. The DSP cannot scan faster than one timer period per data point, or 200 lines/s for an image with 250 points/line in each direction (left-to-right and right-to-left).

The algorithm. Two raster formulas encounter problems when used to generate continuous rasters: recursive algorithms tend to produce cumulative roundoff errors, and integer-based algorithms tend to produce scan size and rate limitations. Thus for programming simplicity we take advantage of our floating-point DSP and use the following formulas (all numbers are floating point).

$$X'_t = X'_0 + t\Delta X',$$

$$Y'_t = Y'_0 + t\Delta Y',$$

where

$$X'_0 = (X_0 - X_c)\cos\theta + X_c - (Y - Y_c)\sin\theta - Y_c,$$

$$Y'_0 = (X_0 - X_c)\sin\theta + X_c + (Y - Y_c)\cos\theta - Y_c,$$

$$\Delta X' = \Delta X \cos\theta,$$

$$\Delta Y' = \Delta X \sin\theta,$$

and (X,Y) = user coordinates, (X',Y') = scanner coordinates, (X_c,Y_c) = center of rotation, ΔX = amount to increment raster with each timer period, X_0 = starting point of scan, t = time in timer periods, $X_t = X$ position at time t, θ = scan rotation. The center of rotation generally coincides with the center of the image.

2. Sample tilt compensation

We have implemented sample tilt compensation as both an enhancement and a substitute for feedback.

With feedback. Adding a raster to Z during scanning tends to improve images of highly sloped surfaces by helping the feedback follow sample tilt. The following recursive algorithm takes about 5 instruction cycles:

$$Z'_t = Z_t + \Delta Z.$$

The DSP inverts the slope ΔZ with each change in scan direction. The PC determines ΔZ with a leveling routine that measures sample height at the corners of the current scan (the software also uses the resulting information to level plotted images). Tilt correction especially helps "fast scan" imaging, in which the user reduces the feedback gains and records the error signal E rather than Z.

Unlike the XY raster, finite word length does not cause problems with the Z raster because the feedback loop can correct for small errors.

Without feedback. Near-field scanning optical microscopes sometimes lack a feedback mechanism. With these instruments, sample tilt compensation in X and Y can help maintain reasonable tip-sample separation while scanning. Instead of the feedback loop, we use the following algorithm to determine Z (all numbers are floating point):

$$Z_t = iX_t + jY_t + k.$$

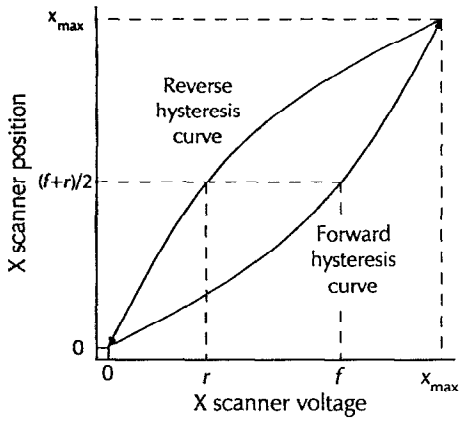

FIG. 7. SPM images show sample features as a function of scanner voltage. Hysteresis correction transforms the images so they show features as a function of scanner position. In our algorithm, the user selects two equivalent points f and r on the forward and reverse images, respectively. The software defines two second-order hysteresis curves to estimate the actual X position as a function of X voltage.

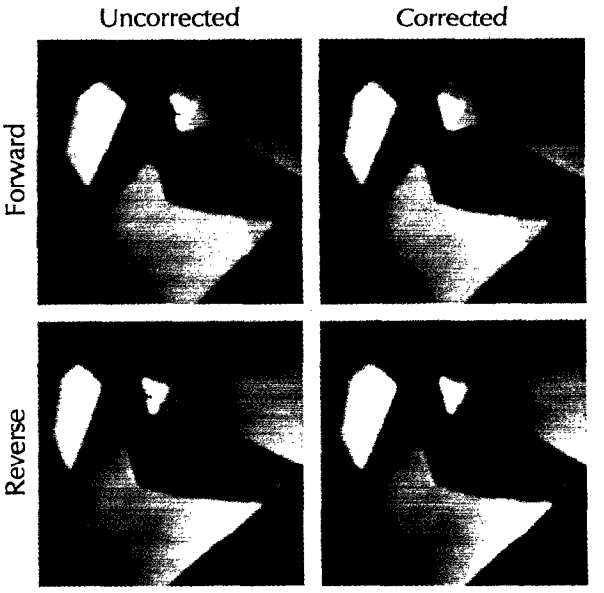

FIG. 8. Example of hysteresis correction. The crosshairs show where the points f and r were chosen. After hysteresis correction the forward and reverse images have a closer resemblance. The image is a 740×740 nm scan of boronated pyrolytic graphite.

The user determines the leveling factors i, j, and k by inspecting image sharpness.

3. Background subtraction

Optical-lever-based AFMs, and our scanned-cantilever instrument in particular, tend to exhibit spurious variations in cantilever deflection signal with the position of the scanner.[18] As a result the downward force that the cantilever exerts on the sample varies with X and Y. With our scanned-cantilever stage, this signal variation arises mostly as a result of cantilever warpage, and the variation with Y position greatly exceeds the variation with X.

We have implemented a background subtraction algorithm that counters the effect of spurious signal variation with Y. The algorithm adds a raster to the setpoint as follows:

$$S' = S + Y\frac{dS}{dY}.$$

A leveling routine determines the slope dS/dY by measuring S at the minimum and maximum Y when the tip is not touching the sample. We use this leveling routine once each time we replace the sample or cantilever.

4. Hysteresis removal

Hysteresis appears because the position of a piezoceramic scanner is not a linear function of the voltage applied to it; rather, position tends to lag behind voltage. For soft piezoceramics the nonlinearity generally measures 25% of the scanner range, while hard piezoceramics have less hysteresis. Although most commercial instruments compensate for hysteresis by generating a nonlinear raster, we have chosen to save processor time by using a simple postacquisition algorithm.

The algorithm warps images by moving their data points according to a user-defined second-order curve (Fig. 7). It assumes that the user has taken both forward (scan direction left-to-right) and reverse (right-to-left) data. The user marks an easily recognized feature on both the forward and reverse images, generating two X coordinates respectively termed f and r (alternatively, the software could determine these coordinates from a scanner calibration). The software warps the images such that f and r both transform to $x=(f+r)/2$ and the edges ($x=0$ and $x=x_{max}$) remain unchanged. It does so by defining a second-order curve from the three points 0, f or r, and x_{max} and making a lookup table according to the following equation (all X coordinates are measured in terms of data points):

$$x = \frac{1-b}{x_{max}}(x')^2 + bx',$$

where

$$b = \frac{x_{max}(f+r) - 2d^2}{2d(x_{max}-d)}$$

and x=old X coordinate, x'=new X coordinate, $d=r$ to transform forward images, $d=f$ to transform reverse images.

The algorithm creates a new image and scans through each point, determining its X coordinate x'. Using this coordinate the algorithm consults the lookup table and reads data from location x in the original image, interpolating when necessary. It then puts this data into the new image at the current location, x'. Figure 8 shows the effect of the algorithm.

If images scanned both bottom-to-top and top-to-bottom are available, the user can repeat the algorithm to remove hysteresis in Y.

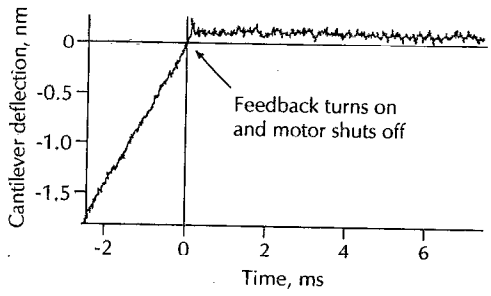

FIG. 9. AFM tip-sample approach curve taken in air. At −2 ms the tip and sample are already in contact and the motion of the approach motor is pushing the cantilever up. When the cantilever deflection reaches its setpoint (zero deflection), the DSP starts the feedback and shuts the motor off. Since the motor takes about 35 ms to come to a halt, the cantilever deflection does not immediately settle to the setpoint.

D. Tip-sample approach

The tip-sample approach routine varies with the approach mechanism. We have used both a continuous approach and a discontinuous approach.

Continuous approach. In a continuous approach, a motor slowly turns a screw that determines tip-sample separation. A DAC voltage determines the motor speed. Generally, before starting the approach, the user has turned the feedback off and set Z at the middle of the scanner range. The engage routine sets the DAC to start the motor and then waits for the sensor signal to cross the setpoint. When it does, the DSP starts the feedback and shuts the motor off.

The software creates a digital oscilloscope display by recording the sensor output C to a circular buffer during the approach. It plots the output for a few milliseconds before and after achieving feedback as a check for false engages and tip crashes (Fig. 9).

Discontinuous approach. In a discontinuous approach, an inchworm, stepper motor, or some other device approaches the tip and sample in well-defined increments, faster than the feedback can respond. The software sets Z such that tip and sample are at their greatest separation before the approach mechanism takes a step. After the step, the software turns the feedback on. If Z goes out of range, tip-sample separation is still too great and the cycle begins again.

E. Modulation measurements

In a modulation image, the DSP measures the dependence of an input on an output at each data point. For example, in STM the dependence of tunneling current on Z (dC/dZ) yields a barrier height image,[20] while the dependence of tunneling current on V gives a density-of-states image.[21] In AFM, the dependence of cantilever deflection on Z (also dC/dZ) yields an elasticity image[22] (Fig. 10). Such modulation images can provide valuable information about sample composition.

In order to take modulation measurements, we have written DSP software that emulates a lock-in amplifier.

Measurement of the first derivative. For lock-in detection of dC/dZ, the DSP adds to Z a sinusoidal modulation

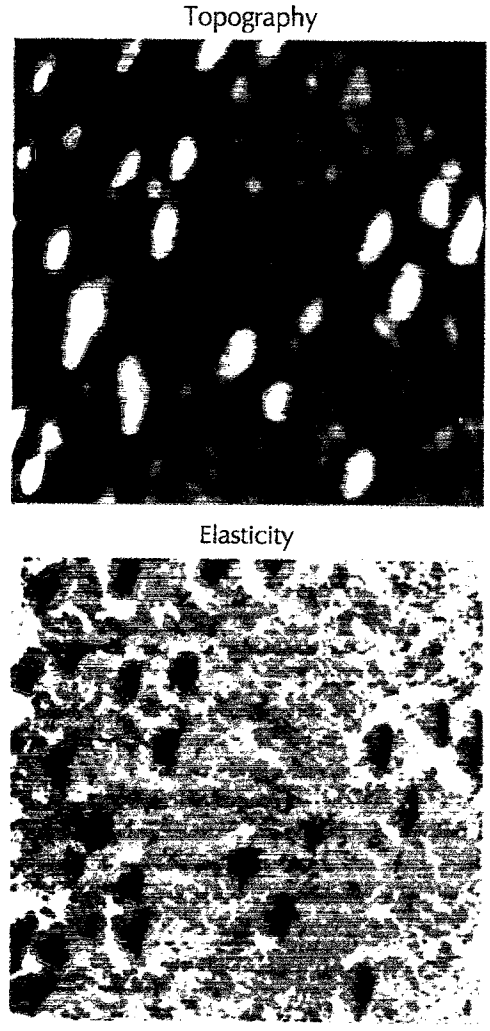

FIG. 10. 1×1 μm AFM modulation (elasticity) image of bovine serum albumen on silicon. Each molecule appears high in the topography image and soft in the modulation image. Modulation amplitude 1 nm.

of an amplitude and frequency chosen by the user (typically, amplitude about 0.1 nm and frequency about 1 kHz). The DSP modulates Z continually, interrupting neither the XY raster nor the feedback to take a modulation measurement:

$$Z_t' = Z_t + Hm_t,$$

where

$$m_t = \sin\left(\frac{2t}{p}\right),$$

H = peak-to-peak modulation amplitude, t = time in timer periods, p = modulation period in timer periods. A circular buffer lookup table contains the values of m_t. The user sets the modulation frequency high enough and the feedback gains low enough so the modulation lies outside the bandwidth of the feedback.

Assuming a linear sensor, the following formulas yield the magnitude of the error signal at the modulation frequency:

$$A_0=\frac{4}{np}\sum_{t=1}^{np} E_t m_t,$$

$$A_{90}=\frac{4}{np}\sum_{t=1}^{np} E_t m_{t+(p/4)},$$

$$A=\sqrt{A_0^2+A_{90}^2},$$

where A=peak-to-peak magnitude of error signal, A_0=amplitude of error signal 0° out of phase with Z modulation, A_{90}=amplitude of error signal 90° out of phase with Z modulation, n=number of modulation periods per data point. The factor $4/np$ provides a peak-to-peak normalization of the amplitudes.

To reduce the demand on the DSP, the DSP stores A_0 and A_{90} without the normalization factors, letting the PC normalize and take the square root. Note that n must be an integer for the algorithm to work; each data point must contain an integral number of modulation periods. For a 1 kHz modulation frequency, the fastest possible scan therefore has one 1 ms modulation period per data point, or 2 lines/s at 250 points/line. Slower rates improve signal to noise.

The above algorithm uses every ADC reading taken during the modulation, giving a better signal-to-noise ratio than differencing algorithms that take only two readings per modulation. In addition, the continuous operation ensures minimal interference with normal imaging. We have found that if the DSP stops the XY raster and feedback for each modulation measurement, a jump or spike appears in Z upon restarting the feedback.

The software should store both A_0 and A_{90} if the application requires both phase and magnitude information. However, it often suffices to record only A_0, since the 90° component provides negligible additional information at low modulation frequencies ($\sim$1 kHz). Not recording the 90° component decreases the processor time required for modulation imaging from 25 to 18 instruction cycles and cuts the amount of data to store in half.

We have found that for high-performance applications the differential nonlinearity of the Z DAC introduces significant noise. For such applications we suggest using separate Z and modulation DACs and summing their outputs with analog electronics, although we have not actually implemented this solution.

Measurement of the second derivative. Current-voltage tunneling spectroscopy can benefit from measurement of the second derivative of current with respect to voltage.[23] The DSP can measure the second derivative simultaneously with the first derivative by detecting at twice the modulation frequency:

$$A_0=\frac{4}{np}\sum_{t=1}^{np} E_t m_{2t},$$

$$A_{90}=\frac{4}{np}\sum_{t=1}^{np} E_t m_{2t+(p/4)}$$

We have used this technique and found it satisfactory.

F. The user interface

We have automated the DSP-based control system using a programmable instruction parser. With this feature a user can devise programs for routine tasks such as approaching the tip and sample or for specific experiments such as taking a series of scans for a movie.

An instruction parser accepts a typed command (like "scan"), carries it out, then prompts for the next command. To such a basic parser we have added an instruction buffer so the user can issue hundreds of instructions at one time; then we added conditional and branch statements, the ability to load sets of instructions (macros) from text files, and the ability to display data and accept user input. For the greatest possible flexibility the instruction set consists of fundamental, low-level operations, effectively forming a SPM programming language.

This programmable parser architecture has many uses. The availability of low-level instructions helps to perform the unusual operations often needed when testing new instruments. For completed instruments we use a "shell" macro that carries out the routine functions of SPM and makes the programmable environment invisible to casual users. We have also developed macros that perform complete experiments under automatic control: e.g., sequences of scans with varying conditions, strings of lithography operations, and area searches. Macros can recognize and rectify common problems such as false engages and use the automatic feedback optimizer to correct poor feedback settings.

The programmable parser offers flexibility but, with a shell macro, does not compromise ease of use. It also simplifies the architecture of the SPM control system by breaking the software into three distinct levels—DSP, PC, and macros.

The authors will gladly provide more detailed information on the software discussed in this article to interested readers.

ACKNOWLEDGMENTS

Financial support was provided by Ford Motor Company, Abbott Laboratories, Topometrix, Inc., a National Science Foundation predoctoral fellowship (D.R.B.), a National Institutes of Health traineeship (S.M.C.), and a Department of Education fellowship (M.G.Y.).

We would like to thank Sie-Ting Wong of Abbott Laboratories for providing the bovine serum albumen sample and Shubert Soares for providing the polished quartz.

[1] O. Marti, S. Gould, and P. K. Hansma, Rev. Sci. Instrum. **59**, 836 (1988).
[2] A. Brown and R. W. Cline, Rev. Sci. Instrum. **61**, 1484 (1990).
[3] R. S. Robinson, T. H. Kimsey, and R. Kimsey, J. Vac. Sci. Technol. B **9**, 631 (1991).
[4] A. J. Hoeven, E. J. van Loenen, P. J. G. M. van Hooft, and K. Oostveen, Rev. Sci. Instrum. **61**, 1668 (1990).
[5] R. Piner and R. Reifenberger, Rev. Sci. Instrum. **60**, 3123 (1989).
[6] S. Park and C. F. Quate, Rev. Sci. Instrum. **58**, 2010 (1987)
[7] A. Hammiche, Y. Wei, I. H. Wilson, and R. P. Webb, Rev. Sci. Instrum. **62**, 3010 (1991).
[8] R. Erlandsson, R. Wigren, and L. Olson, Microsc. Microanal. Microstruct. **1**, 471 (1991).

[9] H. Halling, R. Moeller, and A. Schummers, IEEE Trans. Nucl. Sci. **36**, 634 (1989).
[10] A. Schummers, H. Halling, K. H. Besocke, and G. Cox, J. Vac. Sci. Technol. B **9**, 615 (1991).
[11] D. P. DiLella, J. H. Wandass, R. J. Colton, and C. R. K. Marrian, Rev. Sci. Instrum. **60**, 997 (1989).
[12] S. M. Clark, D. R. Baselt, C. F. Spence, and J. D. Baldeschwieler, Rev. Sci. Instrum. **63**, 4296 (1992).
[13] For a review of scanned-probe experiments, see C. F. Quate, AIP Conf. Proc. **241**, 1 (1992); H. K. Wickramasinghe, *ibid.* **241**, 1 (1992).
[14] A basic system meeting the criteria we outline here might use the Ariel MM-96 DSP board (Ariel Corporation, 433 River Road, Highland Park, NJ 08904; unavailable when we constructed our system), which is compatible with Data Translation analog I/O (Data Translation, 100 Locke Drive, Marlboro, MA 01752), uses one or two floating point DSPs, and supports up to 16 Mb of onboard memory.
[15] Spectrum Signal Processing, Inc., No. 301 3700 Gilmore Way, Burnaby, B.C. V5G4M1, Canada
[16] Texas Instruments, P.O. Box 1443, Houston, TX 77001
[17] Victor J. Bucek, *Control Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
[18] See, for example, R. Isermann, K-H. Lachmann, and D. Matko, *Adaptive Control Systems* (Prentice-Hall, New York, 1992).
[19] D. R. Baselt and J. D. Baldeschwieler, Rev. Sci. Instrum. **64**, 908 (1993).
[20] R. Wiesendanger, L. Eng, H. R. Hidber, and P. Oelhafen, Surf. Sci. **189**, 24 (1987).
[21] M. P. Everson, L. C. Davis, R. C. Jaklevic, and W. Shen, J. Vac. Sci. Technol. B **9**, 891 (1991).
[22] P. Maivald *et al.*, paper presented at Proceedings of Scanning Tunneling Microscopy, 1990 (unpublished).
[23] *Tunneling Spectroscopy*, edited by P. K. Hansma (Plenum, New York, 1982).